ASP.NET Web API 2 開發實戰

建立正確的 RESTful API 開發觀念

多奇數位創意有限公司

技術總監 黃保翕 (Will 保哥)

部落格：http://blog.miniasp.com/

什麼是 REST？

- 全名為 REpresentational State Transfer (表現層狀態移轉)
- 由 Roy Fielding 在 2000 年的博士論文中所提出
  - HTTP 規格 (specification) 的主要編撰者
  - Apache HTTP Server 共同創辦人
- 一種為了**設計分散式系統**而生的**架構風格** (architectural style)
- 不是一種**標準**，而是提供一堆軟體架構設計上的**限制** (constraints)
  - 無狀態 (Stateless)、主從式架構 (Client-Server)、一致性的介面 (Uniform Interface)、...
- 不限於 HTTP 運作環境，但 HTTP 本身就是一個很完整的 REST 實作範例
- 如果一個軟體架構符合 REST 風格，就可稱為 RESTful 架構！

認識 REST 常見的專有名詞

- **資源**（Resources）
  - 網絡上的一個實體，或者說是網絡上的一個具體訊息
  - 它可以是一段文字、一張圖片、一首歌曲、一種 Web 服務
  - 你可以用一個 URI（統一資源定位）指向它，這個 URI 就代表一種 "**資源**"
- **表現**（Representations）
  - 將 **物件** (objects) / **屬性** (attributes) 透過 XML 或 JSON 之類的格式表現出來
- **訊息**（Messages）
  - 表現出來的資料通稱為「訊息」，一般透過 HTTP 方法來取得 (GET, POST, ...)
- **無狀態**（Stateless）
  - 不同的 **要求** (Request) 之間不能在「**伺服器端**」儲存任何用戶端狀態
  - 所有**狀態**必須由「**用戶端**」負責管理 (因為狀態保存會限制架構擴展性)
- **狀態移轉**（State Transfer）
  - 如果 Client 需要取得不同的狀態，就會透過 Server 取得下一個狀態
  - 透過一些 Client 與 Server 之間的互動，產生一種狀態移轉

套用 REST 架構風格的基本條件

- 主從式架構 (Client-Server)
  - 由**客戶端**單方面發起，表現為 Request/Response的形式 (HTTP 就是這樣)
- 無狀態特性 (Stateless)
  - 由**客戶端**負責所有狀態保存 (剛好 HTTP 就是無狀態的)
- 可快取特性 (Cacheable)
  - 回應的內容可以在呼叫的過程中適度快取，以改善執行效率
- 一致性的操作介面 (Uniform Interface)
  - 透過一致的操作介面 (API) 提高 Client/Server 之間互動的可見性
- 分層設計 (Layered System)
  - 透過分層設計，讓不同層級的元件可以分工，或透過負載平衡提高延展性

一致性的操作介面 (Uniform Interface)

- 一個 RESTful Web API 的操作介面
  - 主　　詞：就是 URI (資源位置) (網址)
  - 動　　詞：怎麼操作 (GET, POST, PUT, DELETE, ...)
  - 內容型態：取得的資源格式 (JSON, Text, Image, ...)
- 範例
  - ```
    GET /api/books
    Content-Type: application/json
    ```
  - ```
    POST /api/books
    Content-Type: application/json

    { id: 1, name: "MVC" }
    ```
- 請用 Postman Echo 練習基本 API 操作

RESTful Web API 建議設計方式

- 善用 HTTP 動詞 (Verbs)
- 善用 HTTP 狀態碼 (Status Code) 表達狀態
- 使用 SSL 加密連線
  - Working with SSL in Web API
- 擁有良好的 APIs 文件
- 提供查詢、排序、 篩選功能
- 讓使用者可以決定回傳的欄位
- 使用 JSON 回應訊息

參考來源：http://www.vinaysahni.com/best-practices-for-a-pragmatic-restful-api
進階學習： OData in ASP.NET Web API | The ASP.NET Site

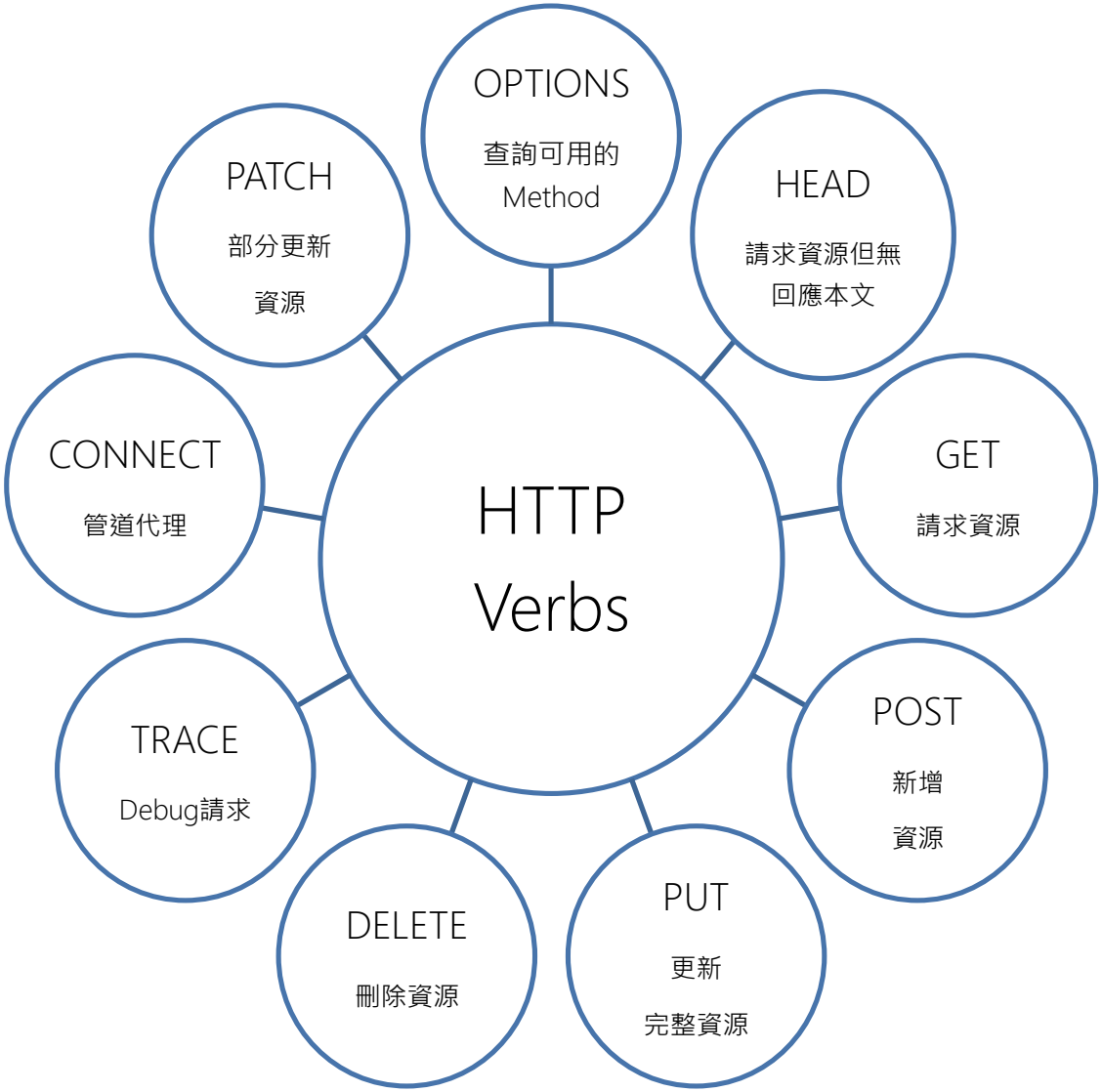
善用 HTTP 動詞 (Verbs)

- 以下兩種說法是相同的意思
  - HTTP Verbs (動詞)
  - HTTP Methods (方法)

- 常見的 HTTP 動詞

  - GET　　取得訊息
  - POST　　建立訊息
  - PUT　　更新完整訊息
  - PATCH　　更新部分訊息
  - DELETE　　刪除訊息

HTTP/1.1: Method Definitions



https://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html

傳統的 API 設計

- 將**動詞**設計**在 URI 網址列**中 (這不能算是 RESTful 風格)
  - `GET /api/getfile`
  - `POST /api/uploadfile`
  - `POST /api/deletefile`

- 回應訊息時**狀態**寫在訊息中 (這也不能算是 RESTful 風格)

```
{
    status: "OK",
    message: "Hello World!"
}
```

RESTful 的 API 設計

- 使用**一致性的操作介面** (Uniform Interface) 表達「資源」
  - `/api/file`

- 將**動詞**設計在 HTTP Verbs 中
  - `GET /api/file`
  - `POST /api/file`
  - `DELETE /api/file`

- 回應訊息時的**狀態**寫在 HTTP **狀態碼**中

```
HTTP/1.1 201 Created
Content-Type: text/json; charset=UTF-8

{
    message: "Hello World!"
}
```

常見 HTTP 狀態碼與主要分類

- 1xx - **參考資訊** (Informational)
- 2xx - **成功** (OK)
  - 200：成功
  - 201：資源已建立
  - 204：處理完成但無回傳資訊
- 3xx - **重新導向** (Redirection)
  - 301：永久轉址
  - 302：暫時轉址
  - 304：未修改
- 4xx - **用戶端錯誤** (Client Error)
  - 400 : 錯誤的請求
  - 401 : 無權限存取
  - 404 : 找不到資源
  - 409 : 請求的處理發生衝突
- 5xx - **伺服器錯誤** (Server Error)
  - 500 : 伺服器發生錯誤

參考文章：
網頁開發人員應了解的 HTTP 狀態碼

參考資料：
HTTP Status Codes

相關圖表：
HTTP Status Codes Art Print by Steve Schoger

適當的回應 HTTP 狀態碼

- Web API 的處理狀態直接使用 HTTP Status Code

功能	URI	HTTP Method	適合的狀態碼建議
新增	/books	POST	201|400|401|409|500
刪除	/books/{id}	DELETE	204|400|401|404|409|500
修改	/books/{id}	PUT	204|400|401|404|409|500
查詢	/books/{id}	GET	200|400|401|404|500
列表	/books	GET	200|400|401|404|500

要有好看好用的 API 文件

- 避免使用 PDF 等格式
- 完整的 Request / Response 週期
- Swashbuckle 5.0 (快速產生文件好物)
  - 快速產生文件
  - 快速產生API測試功能
  - 快速使用

安裝與設定 Swashbuckle

- 安裝 NuGet 套件
  - `Install-Package Swashbuckle`
- 啟用 XML 文件檔案輸出
  - 專案屬性 → 建置 → 輸出 → XML 文件檔案
- 設定 SwaggerConfig 中的 XML 路徑
  - `App_Start\SwaggerConfig.cs`

```
c.IncludeXmlComments(GetXmlCommentsPath());

private static string GetXmlCommentsPath()
{
    return System.String.Format(
                "{0}/App_Data/WebService.XML",
                System.AppDomain.CurrentDomain.BaseDirectory);
}
```

開始使用 Swashbuckle 產生文件

- 開始撰寫 XML 文件註解
  - XML 文件教學課程
  - How to: Use the XML Documentation Features
  - Recommended Tags for Documentation Comments
- 閱讀線上文件
  - http://localhost:25532/swagger/

聯絡資訊

- The Will Will Web

  記載著 Will 在網路世界的學習心得與技術分享

  - http://blog.miniasp.com/
- Will 保哥的技術交流中心 (臉書粉絲專頁) 

  - http://www.facebook.com/will.fans
- Will 保哥的噗浪

  - http://www.plurk.com/willh/invite
- Will 保哥的推特

  - https://twitter.com/Will_Huang